Apparatus and Methods for Entropy-Encoding or Entropy-Decoding using an Initialization of Context Variables

Field of the Invention

The present invention relates to entropy encoding/decoding and, in particular, to entropy encoding/decoding of video signals using a context-based adaptive binary arithmetic coding scheme with initialization

Description of Related Art

Natural camera-view video signals as certain examples for general information signals show, as other information signals, a non-stationary statistical behavior. The statistics of these signals largely depend on the video content and the acquisition process. Traditional concepts of video coding that rely on a mapping from the video signal to a bit stream of variable length-coded syntax elements exploit some of the non-stationary characteristics but certainly not all of them. Moreover, higher-order statistical dependencies on a syntax element level are mostly neglected in existing video coding schemes.

In contrast to the variable length coding, which is also known as Huffman coding, there also exist arithmetic coding schemes, which are mostly binary arithmetic coding schemes for a practical implementation of coding a sequence of information symbols having binary symbols. Such binary symbols are taken from a symbol set which has only two symbols, i.e., a binary "1", and a binary "0". A simple arithmetic coding

scheme is based on a symbol set, which has a certain probability distribution for the occurrence of the respective symbols. In a situation, in which the symbol set is a binary symbol set and the two symbols, i.e., a binary "0" and a binary "1" are equally distributed, the probability information for each symbol equals to 0.5. The probability information for a binary "0" may for example indicate an interval starting from 0 and ending at 0.5, while the probability information for a binary "1" may include an interval starting at 0.5 and ending at 1.0.

The message to be encoded, which is represented by a symbol sequence of information symbols, i.e., a bit stream of zeros and ones is to be represented by a real interval between 0 and 1. Generally, the size of this interval decreases with an increasing message length. However, the number of required bits to indicate the interval increases in this situation. When an arithmetic encoder receives a first symbol, it identifies the interval, to which the first symbol "belongs", i.e., the interval between 0 and 0.5 or the interval between 0.5 and 1.0. Then, when the arithmetic encoder receives the second symbol, it subdivides the interval indicated when encoding the first symbol based on the second symbol. In case the second symbol is again a zero, the newly identified interval will be between 0 and 0.25. In case, however, the second symbol is a binary "1", the second interval will be between 0.25 and 0.5. This procedure is continued until all symbols of the message, i.e., the symbol sequence, are processed. As soon as one determines that a certain digit of the lower or higher bound of an indicated interval thus cannot change any more, which will, for example, be the case, when the interval is determined to lie between for example 0.31 and 0.39, the digit behind the comma, i.e., the "3" can be

output. In binary terms, there can be output the bits that do not change any more, when the interval is further subdivided. Therefore, an arithmetic encoder works as a simultaneous input/output device, i.e., it will output bits although the message to be encoded has not been completely encoded.

In the above example, the assumption was valid that the probability information, i.e., the signal statistics, do not change from symbol to symbol. This assumption restricts the compression gain. Therefore, a context-based adaptive binary arithmetic coding scheme has been developed. This algorithm is described in: D. Marpe, G. Blättermann, and T. Wiegand, "Adaptive codes for H.26L," ITU-T SG16/Q.6 Doc. VCEG-L13, Eibsee, Germany, Jan. 2003-07-10.

Generally, to each symbol to be arithmetically encoded, a context variable specifying a probability model, i.e., probability information is assigned. This probability model is used for arithmetically coding the value of the symbol by a binary arithmetic coding engine.

An important property of arithmetic coding is the possibility to utilize a clean interface between modeling and coding such that in the modeling stage, a model probability distribution is assigned to the given symbols, which then, in the subsequent coding stage, drives the actual coding engine to generate a sequence of bits as a coded representation of the symbols according to the model distribution. Since it is the model that determines the code and its efficiency in the first place, it is of importance to design an adequate model that explores the statistical dependencies to a large degree and that this model is kept "up to date" during encoding.

However, there are significant *model costs* involved by adaptively estimating higher-order conditional probabilities.

Suppose a pre-defined set **T** of past symbols, a so-called *context template*, and a related set **C** = {0,..., C-1} of *contexts* is given, where the contexts are specified by a *modeling function F*. For each symbol x to be coded, a conditional probability $p(x| F(z))$ is estimated by switching between different probability models according to the already coded neighboring symbols $z \in$ **T**. After encoding x, the estimated conditional probability $p(x| F(z))$ is estimated on the fly by tracking the actual source statistics. Since the number of different conditional probabilities to be estimated for an alphabet size of m is high, it is intuitively clear that the model cost, which represents the cost of "learning" the model distribution, is proportional to the number of past symbols to the power of four.

This implies that by increasing the number **C** of different context models, there is a point, where overfitting of the model may occur such that inaccurate estimates of $p(x| F(z))$ will be the result.

This problem can be solved by imposing two severe restrictions on the choice of the context models. First, very limited context templates **T** consisting of a few neighbors of the current symbol to encode are employed such that only a small number of different context models **C** is effectively used.

Secondly, context modeling is restricted to selected bins of the binarized symbols. As a result, the model cost is drastically reduced, even though the ad-hoc design of context mod-

els under these restrictions may not result in the optimal choice with respect to coding efficiency.

Four basic design types of context models can be distinguished. The first type involves a context template with up to two neighboring syntax elements in the past of the current syntax element to encode, where the specific definition of the kind of neighborhood depends on the syntax element. Usually, the specification of this kind of context model for a specific bin is based on a modeling function of the related bin values for the neighboring element to the left and on top of the current syntax element, as shown in Fig. 9.

The second type of context models is only defined for certain data subtypes. For this kind of context models, the values of prior coded bins (b_0, b_1, b_2,…, b_{i-1}) are used for the choice of a model for a given bin with index i. Note that these context models are used to select different models for different internal nodes of a corresponding binary tree.

Both the third and fourth type of context models is applied to residual data only. In contrast to all other types of context models, both types depend on context categories of different block types. Moreover, the third type does not rely on past coded data, but on the position in the scanning path. For the fourth type, modeling functions are specified that involve the evaluation of the accumulated number of encoded (decoded) levels with a specific value prior to the current level bin to encode (decode).

Besides these context models based on conditional probabilities, there are fixed assignments of probability models to bin indices for all those bins that have to be encoded in

regular mode and to which no context model of the previous specified category can be applied.

The above described context modeling is suitable for a video compression engine such as video compression/decompression engines designed in accordance with the presently emerging H.264/AVC video compression standard. To summarize, for each bin of a bin string the context modeling, i.e., the assignment of a context variable, generally depends on the to be processed data type or sub-data type, the position of the binary decision inside the bin string as well as the values of previously coded syntax elements or bins. With the exception of special context variables, the probability model of a context variable is updated after each usage so that the probability model adapts to the actual symbol statistics.

The problem arising from such context-based adaptive binary arithmetic coding engines is the issue what is to be done, when a start symbol of a symbol sequence is to be processed. Here, no previously coded information symbols are present, since the start symbol is the first symbol in the symbol sequence. The straightforward way is to choose a "50 %: 50 %" initialization probability information, which means that the symbol statistics of the start symbol is not known, i.e., the symbol may be zero or may be one in the binary case. In a symbol set with more than two symbols, which is the non-binary case, wherein m is > 2, this corresponds to the use of an equi-probable distribution among all symbols in the symbol set as an initialization probability information. It has been found out that, especially at low bit-rates and/or in connection with relatively small slices, i.e., with relatively small symbol sequences, the rate-distortion performance of the arithmetic coding engine depends on the initialization of

the context modeler. Additionally, it has been found out that depending on certain video sequences, the bit-rate, which is needed to adapt the probability models of the context variables to the actual symbol statistics, can represent a substantial amount of the slice bit-rate. Therefore, especially for low bit-rate applications or relatively short symbol sequences, a degradation of compression efficiency has been discovered.

Summary of the Invention

It is the object of the present invention to provide an improved entropy encoding design with higher compression efficiency.

In accordance with the first aspect of the present invention, this object is achieved by an apparatus for entropy-encoding a symbol sequence of information symbols to obtain entropy-encoded information symbols, the symbol sequence having a start symbol, comprising: an arithmetic encoder for arithmetically encoding a symbol of the symbol sequence based on probability information for the symbol, the symbol being part of a symbol set, to produce the entropy-encoded information symbols; a context modeler for deriving the probability information for the symbol based on a context of the symbol, the context including one or more context symbols processed by the arithmetic encoder prior to processing the symbol, the context modeler including an initializer for initializing the context modeler by determining and providing initialization probability information to the arithmetic encoder, the initialization probability information to be used by the arithmetic encoder for processing the start symbol, wherein the

initializer is operative to determine the initialization probability information based on an estimation of symbol statistics relating to the start symbol such that an initialization probability distribution is different from an equi-probable distribution for all symbols of the symbol set.

In accordance with a second aspect of the present invention, this object is achieved by an apparatus for entropy decoding entropy-encoded information symbols, the entropy-encoded information symbols being produced by arithmetically encoding a symbol of the symbol sequence based on probability information for the symbol, the symbol being part of a symbol set, wherein the probability information for the symbol is derived based on a context of the symbol, the context including one or more context symbols processed earlier, and wherein, for arithmetically encoding the start symbol, an initialization probability information was used, the initialization probability information being based on an estimation of a symbol statistics relating to a start symbol and being determined such that an initialization probability distribution is different from an equi-probable distribution for all symbols of the symbol set, comprising:

an arithmetic decoder for arithmetically decoding the entropy-encoded information symbols to obtain the symbol sequence of information symbols having the start symbol; and

a context modeler for obtaining the probability information used when arithmetically encoding the sequence of information symbols, context modeler including an initializer for obtaining the initialization probability information used when arithmetically encoding the start symbol.

The present invention is based on the finding that an improved compression efficiency can be obtained by initializing the context modeler so that the initialization probability information is different from an equi-probable distribution for all symbols of the symbol set in question. Particularly, the initializer is operative to determine the initialization probability information to be used at least for the start symbol of the symbol sequence based on an estimation of a symbol statistics relating to the start symbol so that the initialization probability information is different from an equi-probable distribution for all symbols of the symbol set.

In particular, this "intelligent" initialization is advantageous for coding of video material with widely varying content from different video sources. It has been found out that, in a preferred embodiment of the present invention, the initializing probability information can be derived based on the quantization parameter used for quantizing the to be encoded information. In video coding applications, and, in particular, in video encoding applications in connection with the newly emerging H.264/AVC video compression standard, the symbol sequence of information symbols to be entropy-encoded is derived from transformed residual values, which are obtained by a time/spatial inter/intra frame prediction mode for example.

Additionally, it has been found out that the intelligent initialization in accordance with the present invention can advantageously be based on prior knowledge on the video source material. In particular, it has been found out that the compression efficiency can be enhanced by defining several sets of initialization values. In accordance with the present invention, for so-called I- or SI-slices, a single set of ini-

tialization values is sufficient. For temporally predictive-coded slices such as the so-called P-, SP- or B-slices, more than one and, preferably, three different sets of initialization values are defined. For these slices, the selected set of initialization values is to be specified by a certain syntax element in the encoder output bit stream so that a decoder processing an encoded output bit stream is able to correctly initialize itself. Preferably, this syntax element is transmitted as part of the slice header. Now, the encoder has the freedom to select the set of initialization values that is the most suitable one for the symbol statistics for each slice.

Preferably, the initialization values are further diversified by detecting, to which data type or sub-data type, the symbol sequence to be processed by the arithmetic encoder belongs to. This certain type is indicated by a context index variable. This context index variable is preferably designed so that it can signal one out of up to 399 different context models from which a corresponding number of initialization probability information can be derived so that an optimally adapted initialization for a large number of different symbol sequences, each of which having a start symbol, can be determined.

It has to be noted, that the to be arithmetically coded symbols have attributed thereto different context models. Preferably, several symbols or bits are to be encoded using one context model out of the plurality of context models. These several symbols or bits form the symbol sequence, wherein the first to be encoded symbol of the symbol sequence, the symbol sequence having attributed thereto a context model, is the start symbol. A context model can include one or even more

than one probability models. In case of a context model having more than one probability models, for each change of probability models, a new initialization will take place.

Brief Description of the Drawings

Preferred embodiments of the present invention are described in detail with respect to the enclosed drawings, in which:

Fig. 1 shows a block diagram of an inventive entropy encoder;

Fig. 2 shows a detailed diagram of the initializer from Fig. 1;

Fig. 3 shows a sequence of steps for calculating a reference index to reference an initialization probability information table including probability information for the least probable symbol (LPS) and a value of the corresponding most probable symbol (MPS) as initialization probability information;

Figs. 4a to 4l show initialization tables referenced by the context index and an initialization index;

Fig. 5 shows a table for illustrating the interrelation of data types on the one hand and initialization tables on the other hand.

Fig. 6 shows a schematic block diagram of an inventive arithmetic decoder;

Fig. 7 shows a block diagram of a complete coding engine including a binarizer and bypass modes for the binarizer and the context modeler;

Fig. 8 shows a basic coding structure for the emerging H.264/AVC video encoder for a macro block;

Fig. 9 illustrates a context template consisting of two neighboring syntax elements A and B to the left and on top of the current syntax element C; and

Fig. 10 shows an illustration of the subdivision of a picture into slices.

Detailed Description of Preferred Embodiments

In the following, reference is made to Fig. 1 for illustrating the structure and functionality of an inventive entropy-encoding apparatus or an entropy-encoding method. In particular, Fig. 1 shows an apparatus for entropy-encoding symbol sequences of information symbols to obtain entropy-encoded information symbols, wherein the symbol sequence has a start symbol. The symbol sequence is input into a symbol sequence input of an arithmetic encoder 12. The entropy-encoded information symbols are output at an entropy-encoder output 14, which is also connected to the arithmetic encoder 12. The arithmetic encoder is operative for arithmetically encoding a symbol of the symbol sequence based on probability information for the symbol. The probability information is input into the arithmetic encoder 12 via a probability information input 16. Since the symbol is part of a specified symbol set, which, in the binary case, only includes two different sym-

bols, i.e., a binary "0" and the binary "1", it is sufficient to provide probability information only for one symbol of the symbol set. In accordance with the present invention, it is preferred to provide the probability information of the least probable symbol (LPS), since this number is naturally smaller than the probability for the most probable symbol. Additionally, the probability information provided to the arithmetic encoder via the probability information input 16 includes the value of the most probable symbol, from which the value of the least probable symbol and vice versa can be easily derived.

The inventive encoder further includes the context modeler 18 for deriving the probability information for the symbol based on a context of the symbol, the context including one or more context symbols input into the context modeler by a context input 20, which have been processed by the arithmetic encoder 12 prior to processing the actual symbol.

In accordance with the present invention, the context modeler 18 includes an initializer 20 for initializing the context modeler by determining and providing initialization probability information to the arithmetic encoder 12, when the initialization probability information is to be used by the arithmetic encoder 12 for processing the start symbol, i.e., the symbol of the symbol sequence, for which no context symbols exist, i.e. symbols which have been processed by the arithmetic encoder 12 prior to the start symbol. This means that, since the arithmetic encoding scheme is a strictly serial scheme, the start symbol is the first symbol of the symbol sequence.

The initializer is operative to determine the initialization probability information based on an estimation of symbol statistics relating to the start symbol such that the initialization probability is different from an equi-probable distribution for all symbols of the symbol set, to which the start symbol belongs to.

In accordance with the present invention, since the context modeler, and, therefore, the arithmetic encoder is initialized in an intelligent way based on an estimation of the symbol statistics that are expected for the symbol sequence to be encoded by the arithmetic encoder, the inventive entropy-encoding apparatus shows an improved start-up performance. Stated in other words, the entropy-encoding apparatus behaves, for the start symbol, as if the start symbol is not the first symbol of a sequence but is an intermediate symbol of a sequence. Therefore, the initialization results in a pre-adaptation of the entropy-encoding apparatus already for the start symbol. Contrary to the straightforward approach, in which the initialization probability information is set to an equi-probable distribution, the arithmetic encoder is immediately context-pre-adapted. In the straightforward case, the context-adaptation would need several steps corresponding to the number of prior encoded symbols forming the context symbols.

It has been found out that the inventive initialization of the context modeler and the arithmetic encoder results, without any further actions, in bit-rate savings of up to 3 % simply by providing an intelligent initialization probability information compared to a straightforward equi-probable distribution.

Fig. 2 shows a preferred embodiment of the initializer 20 from Fig. 1. The initializer 20 includes a retriever for retrieving a quantization parameter for the symbol sequence from a slice header. Additionally, the slice header is also input into a detector for detecting a data type of the slice data. Additionally, a detector 204 for detecting a sub-data type in the slice data is provided. The detector 204 is further operative to access different initialization tables based on the detected sub-data type. As it will be described later on, the access to different initialization tables is controlled by means of a context index on the one hand and, preferably, by means of an initialization index on the other hand. Depending on the slice data, there only exists one initialization table with certain context indices for I- and SI-slices. For P- and B-slices, there exist at least two and preferably three initialization tables which are, once again, addressed by the context index, wherein the table of the at least two tables which is actually used is addressed by the initialization index.

Connected to the retriever 200 on the one hand and the detector 204 on the other hand, is a calculator 206, which is operative to calculate the information on the initialization probability which can be the actual initialization probability or, in accordance with the preferred embodiment of the present invention, a probability state index, i.e., a reference to an initialization probability information table including probability information for the least probable symbol. It has been found out by the inventors that a linear dependence of the initialization probability from the quantization parameter used for quantizing the slice data results in an improved compression efficiency, i.e., in an near to optimum initialization. Therefore, the calculator 206 in Fig. 2

is operative to apply a linear calculation, wherein the parameter m indicates a gradient of the linear dependence, while the other initialization parameter, i.e., n indicates a y-axis offset. The basic equation processed by the calculator 206 in Fig. 2 is indicated in block 206 as

$$m \times QP + n,$$

wherein QP is the quantization parameter while m and n are the initialization parameters taken out from different initialization tables addressed by a certain context index.

In a preferred embodiment of the present invention, the calculator 206 in Fig. 2 performs the sequence of steps as indicated in Fig. 3. In a first step, the equation m x sliceQP + n is calculated. It is to be noted here that in the preferred embodiment of Fig. 3, the value F of m x sliceQP is shifted to the right by four digits. It is to be noted here that, naturally, all values are binary strings. The shift by four digits relates to a representation of the product m x sliceQP in a two-complementary presentation. Then, the result after the shift is added to the other initialization parameter n. Then, the minimum of this result on one hand and the integer value 126 on the other hand is selected. In case this minimum is larger than 1, this minimum is selected and assigned to the auxiliary variable preCtxState. Then, a determination is made, if this auxiliary variable is lower than or equal to 63. In case this determination results in a yes answer, the probability state index variable pStateIdx is calculated as the difference between 63 and the auxiliary variable. Additionally, it is determined that the value of the most probable symbol (valMPS) is equal to zero. In the other case, i.e., in which the auxiliary variable is greater than 63, the

probability state index pStateIdx is calculated as the difference between the auxiliary variable on the one hand and the value of 64. Additionally, the value of the most probable symbol is set to one.

It is to be noted that the probability state index is a reference to an initialization probability information table including probability information for the least probable symbol. The value of the probability state index can address 64 different probability information values in the probability information table. The reason for firstly calculating a probability state index and then addressing a table using the probability state index is in an improved design of the actual arithmetic encoder core. Naturally, by an appropriate mapping, the sequence of the steps shown in Fig. 3 could also directly calculate the initialization probability information.

It is to be noted here that arithmetic encoding is based on the principle of recursive intervals subdivision. Given the probability estimation p(0) and p(1) = 1-p(0) of a binary decision (0,1), an initially given code sub-interval with a certain code range called CODIRange will be subdivided into two sub-intervals having the range p(0) x CODIRange and CODIRange - p(0) x CODIRange, respectively. Depending on the decision, which has been observed, a corresponding sub-interval will be chosen as the new code interval, and the binary code string pointing into that interval will represent the sequence of observed binary decisions. It is useful to distinguish between the most probable symbol (MPS) and the least probable symbol (LPS), so that binary decisions have to be identified as either MPS or LPS, rather than "0" or "1". Given this terminology, each context is specified by the

probability p_{LPS} of the LPS and the value of MPS (valMPS), which is either "0" or "1".

The probability estimation is performed by means of a finite-state machine with a table-based transition process between 64 different representative probability states indicated by the probability state index shown in Fig. 3. For the LPS probability p_{LPS} the number of the states is arranged in such a way that the probability state index pStateIdx = 0 corresponds to an LPS probability value of 0.5, with decreasing LPS probability towards higher state indices.

In the following, reference is made to Fig. 8 to show the complete setup of a video encoder engine including an inventive entropy-encoder as it is shown in Fig. 8 in block 80. In particular, Fig. 8 shows the basic coding structure for the emerging H.264/AVC standard for a macro block. The input video signal is, split into macro blocks, each macro block having 16 x 16 pixels. Then, the association of macro blocks to slice groups and slices is selected, and, then, each macroblock of each slice is processed by the network of operating blocks in Fig. 8. It is to be noted here that an efficient parallel processing of macro blocks is possible, when there are various slices in the picture. The association of macro blocks to slice groups and slices is performed by means of a block called coder control 82 in Fig. 8. There exist several slices, which are defined as follows:

- **I slice**: A slice in which all macroblocks of the slice are coded using intra prediction.

- **P slice**: In addition, to the coding types of the I slice, some macroblocks of the P slice can also be coded

using inter prediction with at most *one* motion-compensated prediction signal per prediction block.

- **B slice**: In addition, to the coding types available in a P slice, some macroblocks of the B slice can also be coded using inter prediction with *two* motion-compensated prediction signals per prediction block.

The above three coding types are very similar to those in previous standards with the exception of the use of reference pictures as described below. The following two coding types for slices are new:

- **SP slice**: A so-called switching P slice that is coded such that efficient switching between different precoded pictures becomes possible.

- **SI slice:** A so-called switching I slice that allows an exact match of a macroblock in an SP slice for random access and error recovery purposes.

Slices are a sequence of macroblocks, which are processed in the order of a raster scan when not using flexible macroblock ordering (FMO). A picture maybe split into one or several slices as shown in Fig. 10. A picture is therefore a collection of one or more slices. Slices are self-contained in the sense that given the active sequence and picture parameter sets, their syntax elements can be parsed from the bitstream and the values of the samples in the area of the picture that the slice represents can be correctly decoded without use of data from other slices provided that utilized reference pictures are identical at encoder and decoder. Some information

from other slices maybe needed to apply the deblocking filter across slice boundaries.

FMO modifies the way how pictures are partitioned into slices and macroblocks by utilizing the concept of *slice groups*. Each slice group is a set of macroblocks defined by a *macroblock to slice group map*, which is specified by the content of the picture parameter set and some information from slice headers. The macroblock to slice group map consists of a slice group identification number for each macroblock in the picture, specifying which slice group the associated macroblock belongs to. Each slice group can be partitioned into one or more slices, such that a slice is a sequence of macroblocks within the same slice group that is processed in the order of a raster scan within the set of macroblocks of a particular slice group. (The case when FMO is not in use can be viewed as the simple special case of FMO in which the whole picture consists of a single slice group.)

Using FMO, a picture can be split into many macroblock-scanning patterns such as interleaved slices, a dispersed macroblock allocation, one or more "foreground" slice groups and a "leftover" slice group, or a checker-board type of mapping.

Each macroblock can be transmitted in one of several coding types depending on the slice-coding type. In all slice-coding types, the following types of intra coding are supported, which are denoted as Intra_4x4 or Intra_16x16 together with chroma prediction and I_PCM prediction modes.

The Intra_4x4 mode is based on predicting each 4x4 luma block separately and is well suited for coding of parts of a pic-

ture with significant detail. The Intra_16x16 mode, on the other hand, does prediction of the whole 16x16 luma block and is more suited for coding very smooth areas of a picture.

In addition, to these two types of luma prediction, a separate chroma prediction is conducted. As an alternative to Intra_4x4 and Intra_16x16, the I_PCM coding type allows the encoder to simply bypass the prediction and transform coding processes and instead directly send the values of the encoded samples. The I_PCM mode serves the following purposes:

1. It allows the encoder to precisely represent the values of the samples

2. It provides a way to accurately represent the values of anomalous picture content without significant data expansion

3. It enables placing a hard limit on the number of bits a decoder must handle for a macroblock without harm to coding efficiency.

In contrast to some previous video coding standards (namely H.263+ and MPEG-4 Visual), where intra prediction has been conducted in the transform domain, intra prediction in H.264/AVC is always conducted in the spatial domain, by referring to neighboring samples of previously coded blocks which are to the left and/or above the block to be predicted. This may incur error propagation in environments with transmission errors that propagate due to motion compensation into inter-coded macroblocks. Therefore, a constrained intra coding mode can be signaled that allows prediction only from intra-coded neighboring macroblocks.

When using the Intra_4x4 mode, each 4x4 block is predicted from spatially neighboring samples. The 16 samples of the 4x4 block are predicted using prior decoded samples in adjacent blocks. For each 4x4 block one of nine prediction modes can be utilized. In addition, to "DC" prediction (where one value is used to predict the entire 4x4 block), eight directional prediction modes are specified. Those modes are suitable to predict directional structures in a picture such as edges at various angles.

In addition, to the intra macroblock coding types, various *predictive* or motion-compensated coding types are specified as P macroblock types. Each P macroblock type corresponds to a specific partition of the macroblock into the block shapes used for motion-compensated prediction. Partitions with luma block sizes of 16x16, 16x8, 8x16, and 8x8 samples are supported by the syntax. In case partitions with 8x8 samples are chosen, one additional syntax element for each 8x8 partition is transmitted. This syntax element specifies whether the corresponding 8x8 partition is further partitioned into partitions of 8x4, 4x8, or 4x4 luma samples and corresponding chroma samples.

The prediction signal for each predictive-coded MxN luma block is obtained by displacing an area of the corresponding reference picture, which is specified by a translational motion vector and a picture reference index. Thus, if the macroblock is coded using four 8x8 partitions and each 8x8 partition is further split into four 4x4 partitions, a maximum of sixteen motion vectors may be transmitted for a single P macroblock.

The quantization parameter SliceQP is used for determining the quantization of transform coefficients in H.264/AVC. The parameter can take 52 values. Theses values are arranged so that an increase of 1 in quantization parameter means an increase of quantization step size by approximately 12 % (an increase of 6 means an increase of quantization step size by exactly a factor of 2). It can be noticed that a change of step size by approximately 12 % also means roughly a reduction of bit rate by approximately 12 %.

The quantized transform coefficients of a block generally are scanned in a zig-zag fashion and transmitted using entropy coding methods. The 2x2 DC coefficients of the chroma component are scanned in raster-scan order. All inverse transform operations in H.264/AVC can be implemented using only additions and bit-shifting operations of 16-bit integer values. Similarly, only 16-bit memory accesses are needed for a good implementation of the forward transform and quantization process in the encoder.

In the following, reference is made to Fig. 7 in order to show a detailed embodiment of the entropy encoder 80 in Fig. 8. In the preferred embodiment of the present invention, the initializer 20 described in connection with Fig. 1 is part of a context modeler 18 in Fig. 7, which is preferably the same device as described in connection with Fig. 1. The context modeler feeds a context model, i.e., a probability information, to an arithmetic encoder 12, which is also referred to as the regular coding engine. In the implementation in Fig. 7, also the to be encoded bit, which is also referred to as a bin, which means binary decision, is also forwarded from the context modeler 18 to the regular coding engine 12. This bin value is also fed back to the context modeler so that a con-

text model update can be obtained which is illustrated by means of feed back line 70. The device shown in Fig. 7 also includes a bypass branch 72, which also includes an arithmetic encoder, which is also called the bypass coding engine 74. The bypass coding engine 74 is operative to arithmetically encode the input bin values. Contrary to the regular coding engine 12 the bypass coding engine 74 is, however, not an adaptive coding engine but works preferably with a fixed probability model without any context adaptation. A selection of the two branches can be obtained by means of switches 76a and 76b as it is shown in Fig. 7. The bypass switch 76a is located between the coding engine branches and a binarizer device 78. The binarizer device 78 is operative to binarize non-binary valued syntax elements for obtaining a bin string, i.e., a string of binary values. In case the syntax element is already a binary value syntax element, the binarizer 78 is bypassed by the bypass branch 79, which is selected by means of a binarizer bypass switch 75.

Fig. 7, therefore, shows the generic block diagram for encoding a single syntax element in CABAC (CABAC = Context-based Adaptive Binary Arithmetic Coding). The encoding process consists of at most three elementary steps:

1. binarization
2. context modeling
3. binary arithmetic coding

In the first step, a given non-binary valued syntax element is uniquely mapped to a binary sequence, a so-called *bin string*. When a binary valued syntax element is given, this initial step is bypassed, as shown in Fig. 7. For each element of the bin string or for each binary valued syntax ele-

ment, one or two subsequent steps may follow depending on the coding mode.

In the co-called *regular coding mode*, prior to the actual arithmetic coding process the given binary decision, which, in the sequel, we will refer to as a *bin*, enters the context modeling stage, where a probability model is selected such that the corresponding choice may depend on previously encoded syntax elements or bins. Then, after the assignment of a context model the bin value along with its associated model is passed to the regular coding engine, where the final stage of arithmetic encoding together with a subsequent model updating takes place (see Fig. 7).

Alternatively, the *bypass coding mode* is chosen for selected bins in order to allow a speedup of the whole encoding (and decoding) process by means of a simplified coding engine without the usage of an explicitly assigned model, as illustrated by the lower right branch of the switch in Fig. 7.

In the following, the three main functional building blocks, which are binarization, context modeling, and binary arithmetic coding, along with their interdependencies are discussed in more detail.

In the following, several details on binary arithmetic coding will be set forth.

Binary arithmetic coding is based on the principles of recursive interval subdivision that involves the following elementary multiplication operation. Suppose that an estimate of the probability $p_{LPS} \in (0, 0.5]$ of the *least probable symbol* (LPS) is given and that the given interval is represented by

its lower bound **L** and its width (range) **R.** Based on that settings, the given interval is subdivided into two sub-intervals: one interval of width

$$R_{LPS} = R \times p_{LPS},$$

which is associated with the LPS, and the dual interval of width $R_{MPS} = R - R_{LPS}$, which is assigned to the most probable symbol (MPS) having a probability estimate of $1 - p_{LPS}$. Depending on the observed binary decision, either identified as the LPS or the MPS, the corresponding sub-interval is then chosen as the new current interval. A binary value pointing into that interval represents the sequence of binary decisions processed so far, whereas the range of the interval corresponds to the product of the probabilities of those binary symbols. Thus, to unambiguously identify that interval and hence the coded sequence of binary decisions, the Shannon lower bound on the entropy of the sequence is asymptotically approximated by using the minimum precision of bits specifying the lower bound of the final interval.

In a practical implementation of binary arithmetic coding the main bottleneck in terms of throughput is the above multiplication operation required to perform the interval subdivision.

In the following, the scheme shown in Fig. 2 will be described in more detail.

The basic self-contained unit in video coding may be a slice. All models are to be re-initialized at the slice boundaries using some pre-defined probability states. In the absence of any prior knowledge about the source, one possible choice

would be to initialize each model with the equi-probable state. However, CABAC provides a built-in mechanism for incorporating some a priori knowledge about the source statistics in the form of appropriate initialization values for each of the probability models. This so-called initialization process for context models allows an adjustment of the initial probability states in CABAC on two levels.

Quantization Parameter Dependent Initialization: On the lower level of adjustment there is a default set of initialization values, which are derived from the initially given slice quantization parameter *Slice*QP, thus providing some kind of pre-adaptation of the initial probability states to the different coding conditions represented by the current value of the *Slice*QP parameter.(ctxIdx = 0 to 398, ctxIdx $\neq$ 276), from which the corresponding *Slice*QP dependent initial probability state is derived during the initialization process by applying the procedure shown in Fig. 3.

Slice Dependent Initialization: The concept of a low-level pre-adaptation of the probability models was generalized by defining two additional sets of context initialization parameters for those probability models specifically used in P and B slices. In this way, the encoder is enabled to choose for these slice types between three initialization tables such that a better fit to different coding scenarios and/or different types of video content can be achieved. This forward adaptation process requires the signaling of the chosen initialization table, which is done by specifying the corresponding table index (0-2) in the slice header. In addition, an increased amount of memory for the storage of the initialization tables is required. However, access to this memory of approx. 3 KB is needed only once per slice. Note that

a chosen initialization table triggers for each probability model the same low-level, *Slice*QP dependent initialization procedure as described in the previous paragraph. Depending on the slice size and the bit-rate, which, in turn, depends on the amount of data that can be used for the backward adaptation process of the symbol statistics, bit-rate savings of up to 3 % have been obtained by using the instrument of slice dependent context initialization.

Figures 4a to 4l show the initialization parameter tables. Fig. 4a for example shows a look up table to be addressed by the context index variable ctxidx as an input address ranging between 0 and 10 in order to retrieve appropriate initialization variables m, n that are to be input into the sequence of steps shown in Fig. 3. Table 1 shown in Fig. 4a is not suitable for P or B slices so that no initialization index or as has been described above, no table index is required. The situation is different in figures 4b, 4c, 4d, 4e, 4g, 4h, 4i, 4j, 4k, 4l.

In all those figures the optimally selected initialization parameters or variables m, n, are not only based on the context index ctxIdx but are also based on a table or initialization index which is also termed as "cabac_init_idc" (CABAC initialization indicator). This means that all those tables where this table index "cabac_init_idc" is present, can be used for coding data in time-domain predicted P, SP and B frames.

Fig. 5 shows an example for assigning different context models to different data types. Data types are slice data, macroblock layer control data, macroblock prediction data, additional prediction data and residual data. All these different

data types can be further diversified into sub-data types so that, for example for the data type residual data, there exist four different sub-data types to which different tables are assigned. Within one sub-data type, there is, preferably, another diversification into different elementary data types, so that, when one elementary data type is detected, a certain pair of initialization parameters m, n, is retrieved by means of a table look up to the tables shown in Fig. 4a to Fig. 4k based on the context index ctxIdx. It is to be noted here that the right four columns in Fig. 5 show the attribution of context index values to data in a certain slice such as slice SI, slice I, slice B, slice SP or slice P.

The inventive initialization based on the a priori known symbol statistics are diversified to a large degree in that up to 398 different initialization probability information items can be calculated. In dependence of the slice quantization parameter and each data type, sub-data type or elementary data type, and also for a slice data type, optimum initialization probability information is calculated to "prime" the entropy encoder so that a maximum bit rate saving can be obtained. In accordance with the invention, bit rate saving losses due to a more or less quick adaptation of the context model because of a non-intelligent initialization are avoided.

In the following, reference is made to Fig. 6 showing an inventive entropy decoder. As it is known in the art of arithmetic coding/decoding, the arithmetic decoder needs the same probability distribution used by the encoder for a respective encoding step to resolve the transmitted interval (which is naturally represented in binary form) into the decoded bits, i.e., the information symbol sequence having the start sym-

bol. To this end, the inventive entropy decoder includes an arithmetic decoder 60 and a context modeler 62, which also includes an initializer 64. It is to be seen here that the decoder shown in Fig. 6 is analogously constructed as the encoder in Fig. 1. Again, the context modeler 62 provides probability information or, when the start symbol of the sequence is concerned, the initialization probability information. This information is produced in the same way as has been described with respect to Fig. 1, i.e., by determining a respective slice quantization parameter, by determining a certain context index and, by determining the corresponding initialization parameters m, n, for calculating the initialization probability information, when the arithmetic decoder is operative to process the start symbol. It is to be noted here that the arithmetic encoding/decoding chain is a kind of first-in-first-out pipeline, since the start symbol of the symbol sequence in the encoder is the first symbol to be encoded and is also the first symbol to be decoded. Therefore, the same initialization process for the entropy decoder shown in Fig. 6 can be used, which has been described in connection with the entropy encoder shown in Fig. 1.

Regarding the regular context adaptation mode, the context modeler also performs the same operational steps as have been described with respect to Fig. 1.

Depending on an actual implementation, the inventive encoding/decoding methods can be implemented in hardware or in software. Therefore, the present invention also relates to a computer program, which can be stored on a computer-readable medium such as a CD, a disk or any other data carrier. The present invention is, therefore, also a computer program having a program code which, when executed on a computer, per-

forms the inventive method of encoding described in connection with Fig. 1 or the inventive method of decoding described in connection with Fig. 6.